Algorithme de Kruskal

1. Arbres et arborescences

Définition : On appelle **arbre** un graphe non orienté **simplement connexe** sans **cycle**.

Propriété : Un arbre possède toujours **n-1 arêtes**. En effet n-1 est le nombre maximum d'arêtes pouvant joindre **n sommets** sans créer de cycle.

D'autre part, s'il possède moins de n-1 arêtes, un graphe ne peut être simplement connexe. car au moins un sommet est isolé.

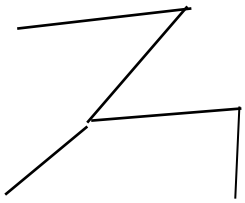
Définition : Dans un graphe orienté, on appelle **racine** un sommet r à partir duquel tous les autres sont accessibles. Si elle existe, la racine d'un arbre orienté est unique.

Définition : On appelle **arborescence** un arbre orienté possédant une (**et une seule) racine**.

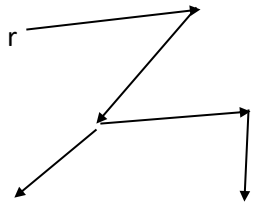
Le graphe du milieu dans la figure ci-dessus est une arborescence de racine r ; celui de droite n'est pas une arborescence car il **n'y a aucune racine**.

Dans un graphe orienté on peut extraire des **sous-graphes partiels** qui constituent des arborescences. Une telle arborescence est dite **maximale** s'il est **impossible** de lui **ajouter** un arc de G sans perdre le caractère d'arborescence.

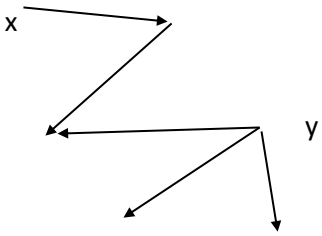
De même, dans un graphe non orienté, on peut extraire des sous-graphes partiels qui constituent des arbres. Un tel arbre est dit maximal s'il est impossible de lui ajouter une arête de G sans perdre le caractère d'arbre.



Arbre (non orienté)



Arbre (orienté) et arborescence de racine r



Arbre (orienté) mais

pas arborescence car x et y ne sont pas accessible l'un à partir de l'autre donc pas de racine

Arbre couvrant minimal :

Dans un graphe non-orienté pondéré G = (V, E, w), un arbre couvrant minimal A est un ensemble d'arêtes de G qui :

i) est un **arbre** (c'est-à-dire **connexe et acyclique**)
ii) touche tous les sommets de G (c'est "**couvrant**")
iii) est de **poids minimal**, c'est-à-dire $\sum_{\{u,v\}\in A} w(u,v)$ que est minimal

Algorithme de Kruskal :

L'algorithme de Kruskal permet de **déterminer l'arbre couvrant de poids minimal** dans un graphe pondéré positivement. Il est utilisé pour la **communication multicast** et implémenté par plusieurs algorithmes de routage (exemple **EKRUS**). Son principe de base est de représenter les classes de connexité par des ensembles. Au départ, on supprime toutes les arêtes, puis on ajoute celles de poids le plus petit possible nécessaire à unir tous les sommets dans une unique classe de connexité.

Dans un graphe non-orienté pondéré $G = (V; E; w)$, un arbre couvrant minimal A est un ensemble
d'arêtes de G qui :
i) est un arbre (c'est-`a-dire connexe et acyclique)
ii) touche tous les sommets de G (c'est couvrant)
iii) est de poids minimal, c'est-`a-dire que X

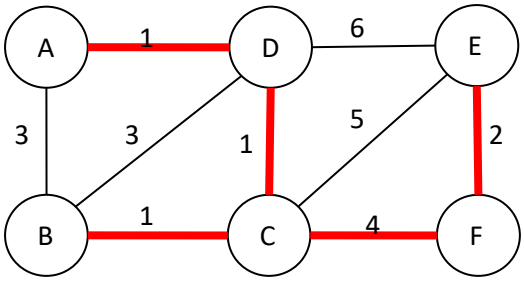
$w(u; v)$ est minimal

Ceci se traduit par l'algorithme suivant :

```
2:    A' ← ∅
3:    P ← ∅
4:    for each x ∈ V do MakeSet(P, x)
5:    trier E par ordre croissant de poids w
6:    for all {s, t} ∈ E (dans l'ordre) do
7:        if Find(P, s) ≠ Find(P, t) then
8:            A' ← A' ∪ {{s, t}}
9:            Union(P, s, t)
10:       end if
11:   end for
```

i) MakeSet(P, x) : créée dans P un ensemble de singleton contenant x.
ii) Union(P, x, y) : indique qu'en fait x et y appartiennent au même ensemble.
iii) Find(P, x) : renvoie le représentant de l'ensemble contenant x.

Exemple : trouver l'arbre couvrant minimale du graphe G (V, E, w) en utilisant la méthode de KRUSKAL


A
D
E
B
C
F
1
6
3
3
1
5
2
1
4